PÁRHUZAMOS PROGRAMOZÁSI MÓDSZERTAN

Horváth Zoltán, *hz@lngsc2.inf.elte.hu*
Eötvös Loránd Tudományegyetem
Általános Számítástudományi Tanszék

Abstract

This paper concerned with a parallel programming methodology used in the education of computer science at University Eötvös, Budapest.We introduce the basic concepts of a relational model of parallel programming. We define the concepts of a problem, an abstract program and a solution. Our approach is functional, problems are given an own semantical meaning. The abstract program is regarded as a relation generated by a set of nondeterministic conditional assignments similar to the concept of abstract program in UNITY. We introduce the behaviour relation of a parallel program which is easy to compare to the relation which is the interpretation of a problem. The synthetized abstract programs are implemented on a parallel computer using C and PVM.

Az ELTE programozó matematikus szakon az 1993/94-es tanévtõl indult meg a *Párhuzamos programozás alapjai* c. tárgy oktatása. A tárgy keretében egy olyan *relációs programozási modell* alapfogalmait ismertetjük, amely alkalmas arra, hogy párhuzamos prog ramok szintetizálását támogassa. A bemutatott modell két fontos elõzményre épít. Az egyik a nemdeterminisztikus szekvenciális prog ramok reláció alapú modellje, amelyet Fóthi Ákos dolgozott ki az ELTÉ-n [3,12], a másik Chandy és Misra párhuzamos prog ramozási módszertana [1].

A tárgy tematikájának kidolgozása során figyelembe vettük a hallgatók programozási elõismereteit, így támaszkodunk a szekvenciális prog ramok reláció alapú modelljére, valamint a Programozási nyelvek tárgy keretében szerzett elemi párhuzamos programozási ismeretekre. Annak tudatában, hogy párhuzamos programok utólagos helyességbizonyítása, de különösen hibás programok javítása, tesztelése rendkívül idõigényes feladat, ezért egy olyan egyszerû matematikai eszközkészlet kialakítására törekedtünk, amely lehetõvé teszi a feladat specifikációjának ismeretében bizonyíthatóan helyes megoldás lépésenkénti elõállítását. A kapott eredményeket a hallgatók a gyakorlatban sikeresen alkalmazzák, a tanult módszerekkel három-három párhuzamos programot készítenek el valódi párhuzamos környezetben a félév során.

Gondolkodásunk során a programozási feladat [3] fogalmából indulunk ki. A feladatok specifikációit lépésenként finomítjuk. Az elsõ specifikáció rövid, csak a leg lényegesebb elemeket tartalmazza. A specifikációs lépések során arra törekszünk, hogy a hatékonysági szempontok, a konkrét számítógép architektú-ra speciális jellemzõi minél késõbb jelenjenek meg a megoldás kidolgozása során. Ál talában az architektúrától, a konkrét ütemezési stra tégiától, a használt programozási nyelvtõl független megoldáshoz, az *absztrakt programhoz* jutunk el. Az abszt rakt program implementációját szabványos *leképezési módszerek* alkalmazásával készítjük el. A megoldás bonyolultságát, a hatékonyság kérdését a konkrét, leképezett program esetében vizsgáljuk. A megoldás tervezése során arra törekszünk, hogy a program kis döntések sorozata eredményeként álljon elõ, hogy minden egyes lépés helyessége könnyen belátható legyen.

A modell ismertetése során törekszünk arra, hogy lehetõség szerint az új fogalmakat formális definíció helyett inkább szövegesen adjuk meg. A modell alkalmazásához azonban elkerülhetetlen a formális definíciók ismerete is [7,8,10].

1. A programozási feladat

A relációs programozási modellben az - általánosított - programozási feladat és a feladat finomításának fogalma a modell legfontosabb elemei közé tartozik. A modell eszközeinek segítségével a feladat specifikációját helyettesíteni tudjuk olyan feladatok specifikációival, amely feladatok megoldása esetén *levezetési szabályokkal* belátható az eredeti feladat megoldásának helyessége [1,8,10]. A lépésenkénti finomítás során tehát magát a feladatot és nem a programot finomítjuk. A feladatnak önálló szemantikai jelentése van, így a lépésenkénti finomítás a feladatok felett értelmezett reláció.

A feladat definíciójának megfogalmazásakor ál talánosítjuk azt a specifikációs módszert, amely relációk segítségével megfogalmazott elõ- és utófeltételeket használ. A bevezetett feladatfogalom magában foglalja azt az esetet is, amikor egy vagy több nem feltétlenül terminál folyamat, egy zárt rendszer együttes viselkedésére teszünk elõírásokat. Megjegyezzük hogy a feladat függetlenül megfogalmazható bármely lehetséges megoldásától, összehasonlítható más feladatokkal, illetve összevethetõ tet szõleges vele közös állapottéren futó programmal abból a szempontból, hogy az megoldja-e. A feladat megoldása nem feltétlenül csak párhuzamos program lehet.

Programozási feladatot mindig egy állapottéren [2] fogalmazunk meg. A feladat megfogalmazásához tehát modellt kell alkotnunk, absztrakcióra van szükség.

Az állapottér véges sok legfeljebb megszámlálhatóan végtelen típus értékhalmaz direkt szorzata [3]. Ha az állapottér például három típusértékhalmaz direkt szorzataként állt elõ, akkor az állapotok adott típusú értékek rendezett hármasai. A rendezett hármas egy-egy eleme definiálja egy-egy változó értékét az adott állapotban. A változók tehát az állapottéren értelmezett projekciós függvények. Relációnak nevezzük egy direkt szorzat részhalmazait. Bináris relációról beszélünk, ha a direkt szorzat kétkomponensû. Egy bináris reláció értelmezési tartományát azon pontok alkotják, amelyek elõfordulnak a relációt alkotó rendezett párok elsõ tagjaként. Logikai függvénynek nevezzük egy bináris relációt, ha függvény és értékkészlete a logikai értékek halmaza [3,8,12].

A feladat matematikai megfelelõje *specifikációs relációk* együttese. A specifikációs relációkat az állapottér hatványhalmaza felett értel mezzük, a relációk elemeit *specifikációs feltételeknek* nevezzük. A relációk segítségével megfogalmazhatunk például olyan kikötéseket, hogy a program $P \wedge \neg Q$ logikai függvény igazsághalmazából Q igazsághalmazának érintése nélkül közvetlenül $\neg Q \wedge \neg P$ állítás igazsághalmazába jusson.

1.1. Specifikációs feltételek

A specifikációs feltételek a programra, mint az állapottér feletti mozgásra fogalmaznak meg kikötéseket. Ezeket a kikötéseket csoportosíthatjuk típusuk szerint. Egy feladat leírásához hét féle feltételt használunk. Az azonos feltételtípushoz tartozó feltételeket egy relá cióban gyûjtjük össze, így hét specifikációs relációt vezetünk be.

- A $P > Q$ és az $inv\ P$ alakú feltételeket *biztonságossági feltételeknek* nevezzük. Ha a program állapotára teljesül a $P \wedge \neg Q$ feltétel, akkor $P > Q$ tiltja, hogy a program Q érintése nélkül közvetlenül egy $\neg Q \wedge \neg P$ -beli állapotba jusson. $inv\ P$ pedig kiköti, hogy a P feltétel igazsághalmazából a program minden elemi lépése a P igazsághalmazába vigyen, valamint, hogy P "kezdetben" is teljesüljön.

- A P a Q, illetve a $P \longrightarrow Q$ *haladási feltételek* elõírják, hogy ha a program egy P-beli állapotba jut, akkor abból elõbb - utóbb Q -ba jusson. P a Q további megszorítást tesz a haladási irányra. $Q \longrightarrow FP$ kikötésnek megfelelõ program elõbb-utóbb *biztosan fixpontba jut* Q -beli állapotából.

- A $FP \Rightarrow R$ *fixpont feltételekkel* szükséges feltételeket fogalmazunk meg arra, hogy a program fixpontba jusson.

- Elégségesnek tekintjük, ha $Q \in INIT$ kezdeti feltételekkel meghatározott állapotokból indítva helyesen mûködik a program.

1.2. A programozási feladat definíciója

Legyen A egy állapottér, B pedig egy tetszõleges, megszámlálható halmaz. Rendeljünk hozzá a $b \in B$ pontokhoz rendezett reláció heteseket. Minden egyes rendezett hetes kettõ, peremfeltételeket megadó, illetve öt, átmenetfeltételeket leíró relációt tartalmaz. Az így kapott relációt az A állapottér felett definiált *feladatnak*, B -t pedig a feladat *paramétertere*nek nevezzük. A $b \in B$-hez rendelt $h \in F(b)$ rendezett relációhetes komponenseit rendre $>_h$ a $_{h,} \longrightarrow_h$, TERM $_h$, FP $_h$, inv $_h$, INIT $_h$ -val jelöljük. Ha F(b) egyelemû, akkor h helyett b-t írunk vagy a h indexet teljesen elhagyjuk, ha ez nem okoz félreértést.

A paramétertér helyes megválasztásával elérhetjük, hogy a $>_h$, a $_h$, stb. relációk végesek legyenek, vagy éppen csak egyetlen egy halmazpár legyen az elemük. Ha a B paramétertér végtelen, akkor így összességében végtelen sok relációt adunk meg. Ezek a relációk azonban általában csak a b paraméter értékében különböznek egymástól, így a megoldás definícióját elegendõ lesz egyetlen paraméteres esetre megvizsgálni.

A paramétertér bevezetésével könnyen megfogalmazhatunk olyan feladatokat, amelynek megoldása több alternatív viselkedésminta szerint is lehetséges.

2. Absztrakt párhuzamos program

Az *absztrakt párhuzamos program* a van Lamsweerde és Sintzoff által definiált [11] és a UNITY-bõl [1] is ismert iteratív programstruktúra relációs alapú megfogalmazása. Ahhoz, hogy eldöntsük, hogy egy program *megold-e* egy feladatot, össze kell vetnünk a feladatot definiáló relációt a *program viselkedési relációjával*.

Az absztrakt program struktúrája nem eredményezheti, hogy olyan szinkronizációs kényszerek épüljenek be a megoldásba, amelyek feleslegesek, valódi párhuzamos architektúrán szükségtelenül lassítják a program futását. Ha a programot szekvenciális folyamatok halmazának tekintenénk, ahogy ezt pl. CSP-ben vagy Adában megszoktuk, akkor ezzel eleve végrehajtási sorrendet definiálnánk utasítások nagy részhalmazai felett. Ezért a párhuzamos programot utasítások halmaza segítségével definiáljuk. A végrehajtás minden egyes lépése során kiválasztunk egy utasítást. A nemdeterminisztikus végrehajtási sorrend korlátozására szinkronizációs feltételeket használunk (pl. termelõ-fogyasztó esetén üres pufferbõl nem lehet fogyasztani). Feltételezzük, ha több processzor hajtja végre a konkrét programot, akkor ez hatékonysági szempontoktól eltekintve hatásában megegyezik azzal, mintha egyetlen processzor válogatott volna valamilyen nemdeterminisztikus sorrendben az utasításhalmaz elemei közül. Megengedjük ugyan, hogy két vagy több processzor idõben átfedve hajtsa végre ugyanazon utasítás különbözõ elemi lépéseit vagy különbözõ utasításokat, de az így kapott eredménynek meg kell egyeznie valamelyik eredménnyel azok közül, amelyet valamelyik végrehajtási út mentén egyetlen processzor állított volna elõ [1]. Feltételezzük tehát, hogy az absztrakt programot oly módon implementáljuk, hogy elemi építõköveire, az utasításokra párhuzamos végrehajtás esetén teljesül a *sorbarendezhetõség* követelménye. Egy-egy sorrendet egy-egy végrehajtási út ír le.

Az utasításokat szekvenciális programokként definiáljuk [3,12]. Egy szekvenciális programot, mint az állapottér feletti mozgást állapotsorozatokkal írhatunk le. Az utasítás tehát egy olyan reláció, amely az állapottér pontjaihoz az állapottér pontjaiból alkotott véges vagy végtelen sorozatokat rendel [3]. Számunkra most elsõsorban az utasítás hatása érdekes, ezért bevezetjük a hatásreláció fogalmát. Egy utasítás hatásrelációja az állapottér azon pontjaiban értelmezett, amelyekhez az utasítás kizárólag véges sorozatokat rendel. A hatásreláció egy az értelmezési tartományába tartozó állapothoz, az utasítás által az állapothoz rendelt sorozatok végpontjait rendeli hozzá [3]. Egy-egy értékadást annak hatásával jellemezhetünk. Megadjuk, hogy az értékadás végrehajtása esetén milyen állapotváltozás következik be. Az értékadás egyszerre több változó értékét is meg változtathatja, ezért ún. szimultán értékadásról van szó [12].

Az absztrakt program utasításait szimultán feltételes értékadások [1,7,8] for májában adjuk meg, ahol az egyes értékadások jobb oldalán függvénykompozíciók is szerepelhetnek (pl. kapcsos zárójellel megadott függvény).

Az absztrakt programot egy olyan bináris relációként definiáljuk, amelyik egy kezdeti feltételes értékadás hatásrelációja, illetve véges sok feltételes értékadás hatás relációjának diszjunkt uniója által generált fák ekvivalenciaosztályait rendeli az állapottér egyes pontjaihoz [8].

2.1. A pártatlan ütemezés fogalma

A megoldás definíciójában kimondjuk, hogy a feladatban megfogalmazott feltételeknek csak azokra a végrehajtási utakra kell teljesülniük, amelyek mentén a feltételes értékadások halmazából minden utasítás végtelen sokszor kerül kiválasztásra. Ez azt jelenti, hogy az absztrakt program implementációjának elkészítésekor a program helyes mûködése érdekében biztosítani kell a feltétlenül pártatlan ütemezést [1,8].

2.2. Modulok és folyamatok

Az utasítások halmazát sok esetben halmazmûveletek segítségével ál lítjuk majd elõ a megoldás logikai struktúrájának megfelelõ modulokból. Egy-egy modul leírhatja például egy-egy objektum viselkedését [1]. A modulok uniójaként vagy szuperpozíciójaként kapott program [1,10] utasításait hatékonysági szempontok figyelembevételével ké pezhetjük le logikai vagy fizikai processzorokra. A modul tehát programtervezési, a folyamat pedig implementációs fogalom.

2.3. Az absztrakt program tulajdonságai

Az absztrakt programok tulajdonságait az állapottér hatványhalmaza felett értelmezett relációkkal írjuk le.

Az absztrakt programok jellemzésekor támaszko dunk a leggyen gébb elõfeltétel [2], a legszigorúbb utófeltétel fogalmára. Azt az állítást, amelynek igazsághalmaza megegyezik azon állapotok halmazával, amelyekbõl egy adott s utasítás végrehajtásával biztosan eljutunk egy elõre megadott R állítás igazsághalmazába $wp(s,R)$ -rel jelöljük, és az R állítás s utasításra vonatkozó leggyengébb elõfeltételének nevezzük [2]. $[wp(s,R)] = \{a \in D_{p(s)} \mid p(s)(a) \subseteq [R]\}$.

Hasonlóképpen a Q állítás s-re vonatkozó legszigorúbb utófeltétele, $sp(s,Q)$ azon pontok halmazát írja le, amelybe a Q igazsághalmazából az s végrehajtásával eljuthatunk.

$inv_S(Q)$-val jelöljük azon P logikai függvények igazsághalmazainak halmazát, amelyek az S programra nézve invariánsok, ha a program Q-beli állapotból indul.

$>_S$ -sel jelöljük azon P,Q logikai függvények igazsághalmazai rendezett párjainak halmazát, amelyekre az S program végrehajtása során igaz, hogy P stabil feltéve, hogy nem Q.

a_S -sel jelöljük azon P,Q logikai függvények igazsághalmazai rendezett párjainak halmazát, amelyekre az S program végrehajtása során igaz, hogy P stabil feltéve, hogy nem Q és van egy olyan s_j eleme S

feltételes értékadás, amely garantálja, hogy a P -bõl Q -ba jutunk. Legyen $\longrightarrow_S$ a a $_S$ reláció tranzitív diszjunktív lezártja.

Terminálnak tekintünk egy izolált programot akkor, ha elérte egy fixpontját, azaz a további mûveletek hatására állapotváltozás már nem következik be. Jelöljük fixpont $_S$ -sel azt a logikai függvényt, amelynek igazsághalmaza megegyezik az S program fixpontjainak halmazával. fixpont $_S$ az S -ben elõ forduló feltételes értékadásokból egyszerûen kiszámolható [1,8].

Jelöljük FP_S -sel azon R logikai függvények igazsághalmazainak halmazát, amelyekre fixpont$_S \Rightarrow$ R . $TERM_S$ -sel jelöljük azon Q logikai függvények igaz sághalmazainak halmazát, amelyekre Q $\longrightarrow_S$ fixpont$_S$.

Egy program szemantikai jelentését azonosíthatjuk az általa az állapot tér hatványhalmaza felett definiált - invariánsok, biztonságossági, haladási, fixpont és terminálási tulajdonságoknak megfelelõ unáris és bináris - relációk együttesével, a program *viselkedési relációjával* . Ez a szemantika leíró jellegû.

3. Megoldás

A megoldás fogalmát a leggyengébb elõfeltétel fogalmára építjük fel. A megoldás definíciója így egy olyan verifikációs kalkulus alapja, amely helyes program esetén a specifikációs feltételek és az absztrakt program utasításainak számával lineárisan arányos számú lépésben véget ér. Azt mondjuk, hogy az S program megoldja az F feladatot, ha a paramétertér minden b eleméhez található olyan $h \in F(b)$, hogy az S program megfelel a h -ban adott P $>_h$ Q, P a $_h$ Q, P $\longrightarrow_h$ Q, Q $\in$ TERM $_h$, FP $_h \Rightarrow$R , $P \in$ inv $_h$ alakú specifikációs feltételek mindegyikének a Q $\in$ INIT $_h$ kezdeti feltételek mellett. Rögzített program esetén indokolt a specifikációs fel tételek vizsgálatát az elérhetõ állapotok halmazára korlátozni. Ha a program megfelel egy specifikációs feltételnek az elérhetõ állapotok felett, akkor a specifikációs feltétel nem sérül a program futása során [7,8]. A gyakorlatban azonban általában az elérhetõ állapotok halmazánál tágabb halmazt választunk, szélsõ esetben akár az összes állapotot, a teljes állapotteret is figyelembe vehetjük.

4. Programszintézis

Megadtuk a prog ramozási feladat, az absztrakt prog ram, az absztrakt prog ram viselkedési relációja és a megoldás definícióját [8]. A programozási feladatok megfogalmazására és megoldására koráb ban sikeresen alkalmazott módszereket [2,12] párhuzamos programokra is alkalmazhatjuk. A megközelítés funkcionális, más hasonló párhuzamos programozási modellektõl eltérõen a feladatnak önálló szemantikai jelentése van, így a lépésenkénti finomítás a feladatok és nem a programok felett értelmezett reláció. Az absztrakt program és tulajdonságai a temporális logikával rokon UNITY logikából [1] ismert programfogalom relációs alapú megfogalmazásai.

Az ismertetett modellben formális eszközökkel ellenõrizhetõ, hogy egy specifikáció finomítása-e egy másiknak, illetve, hogy egy program megoldása-e egy feladatnak. A megoldás definíciója végsõ soron a leggyengébb elõfeltétel fogalmára épül. Az absztrakt program értékadások halmaza. Értékadások leggyengébb elõfeltételét könnyen meghatározhatjuk, így a modell alkalmazásakor az egyes lépések helyessége formális módszerekkel is viszonylag egyszerûen elvégezhetõ.

Formálisan definiáljuk programok szekvenciáját, bevezetjük a UNITY-bõl ismert programkonstrukciókat, az uniót és a szuperpozíciót [8,10]. Kimondunk levezetési sza bályokat, amelyek segítségével a lépésenkénti finomítás során kapott feladatok megoldása után az eredeti feladat megoldását könnyen megadhatjuk.

Mindezek felhasználásával programozási tételeket vezetünk le (asszociatív mûvelet eredménye [6], párhuzamos elemenkénti feldolgozás [5], adat csatorna tétele, stb.) majd konkrét feladatok megoldása során alkalmazzuk a tételeket.

Az asszociatív mûvelet eredményeinek párhuzamos ki számítására levezetett tétel az egyik leggyakrabban elõforduló feladatosztály esetére nyújt aszinkron architektúrán is ha tékonyan implementálható, verifikált megoldást. Elemenként feldolgozható fü ggvények eredményének párhuzamos kiszámítására alkalmas programozási tételt nagy adatfeldolgozási feladatok párhuzamos megoldásakor alkalmazhatjuk.

A kapott absztrakt prog ramokat egy PowerXplorer típusú, 16 processzoros, pár huzamos számí tógépen implementáljuk C nyelven, PVM modellben. Vizs gáljuk az implementált program hatékonyságát.

A tárgy keretében elsajátított elméleti és gyakorlati párhuzamos programozási ismereteket hallgatóink az Operációs rendszerek és a Programozási módszertan elmélete tárgy, ill. a Párhuzamos folyamatok sáv keretében mélyíthetik el.

Irodalomjegyzék

[1] Chandy, K. M., Misra, J.: *Parallel program design: a foundation* , Addison-Wesley, 1989.

[2] Dijkstra, E. W.: *A Discipline of Programming* , Prentice-Hall, 1976.

[3] Fóthi Á.: A Mathematical Approach to Programming . *Annales Uni. Sci. Budapest de R. Eötvös, Nom. Sectio Computatorica* , Tom. IX. (1988) 105-114.

[4] Fóthi Á.-Hunyadvári L.: Programozási módszertan az ELTE programozó matematikus képzésében. *Informatika a felsõoktatásban* , Országos konferencia, Debrecen, 1993. szeptember 1-3. 196-201.

[5] Fóthi Á.- Horváth Z.- Kozsik T.: Parallel Elementwise Processing – A Novel Version. Varga L., ed., *Proceedings of the Fourth Symposium on Programming Languages and Software Tools* , Visegrád, Hungary, June 9-10, 1995 (1995) 180-194. (to appear in: *Annales Uni. Sci. Budapest de R. Eötvös, Nom. Sectio Computatorica.)*

[6] Horváth Z.: Parallel asynchronous computation of the values of an associative function. *Acta Cybernetica* , Vol. 12, No. 1, Szeged (1995) 83-94.

[7] Horváth Z.: The Formal Specification of a Problem Solved by a Parallel Program – a Relational Model.Varga L., ed., *Proceedings of the Fourth Symposium on Programming Languages and Software Tools* , Visegrád, Hungary, June 9-10, 1995 (1995) 165-179. (to appear in: *Annales Uni. Sci. Budapest de R. Eötvös, Nom. Sectio Computatorica.)*

[8] Horváth Z.: *Párhuzamos programok relációs programozási modellje* , PhD dolgozat, ELTE, TTK, Informatika Doktori Iskola, 1996.

[9] Horváth Z.- Kozma L.: Parallel Programming Methodology. In: Bogdany J.-Vesztergombi G., ed., *Workshop on Parallel Processing* . Technology and Applications. Budapest, Hungary, 10-11 February, 1994, KFKI-94-09/M,N Report (1994) 57-65.

[10] Horváth Z., Kozsik T., Venczel T.: On Composing Problems and Parallel Programs, submitted to: *DAPSYS'96* , Miskolc, Hungary, 1996.

[11] van Lamswerde , A., Sintzoff, M. : Formal Derivation of Strongly Correct Concurrent Programs. *Acta Informatica* , Vol. 12, No. 1 (1979) 1-31.

[12] WRMP - Fóthi Á. et al.: Some Concepts of a Relational Model of Programming, in: Varga L., ed., *Proceedings of the Fourth Symposium on Programming Languages and Software Tools* , Visegrád, Hungary, June 9-10, 1995 (1995) 434-446.